Državno natjecanje iz informatike

Srednja škola

Druga podskupina (3. i 4. razred)

20. i 21. ožujka 2019.

OPISI ALGORITAMA

Autori zadataka: Domagoj Bradač, Mislav Bradač, Adrian Satja Kurdija, Tonko Sabolčec

Eventualna pitanja, mišljenja ili prijedloge vezane uz zadatke uputite na adrian.kurdija@fer.hr.

Ministarstvo znanosti i obrazovanja

Agencija za odgoj i obrazovanje

HRVATSKI SAVEZ INFORMATIČARA

Zadatak možemo riješiti na više načina. Jedan je način nacrtati sve bridove "zadatak - početno slovo" i koristiti standardni algoritam za uparivanje (bipartite matching). Lakši je način rekurzivno generirati i provjeriti sve moguće odabire, kojih je dovoljno malo, najviše 4^{12}.

Testne primjere vrijedne 50% bodova moglo se riješiti simulacijom postupka opisanog u zadatku. Da bismo riješili ostale testne primjere, bilo je potrebno primijetiti da ne moramo u svakom koraku prolaziti cijelom tablicu kako bismo detektirali koji se redci i stupci brišu te da nije potrebno doslovno provoditi brisanje ćelija iz tablice.

U prvom koraku potrebno je proći cijelom tablicom kako bismo pronašli retke i stupce za brisanje, a u ostalim koracima moramo provjeriti samo ćelije koje su u neposrednoj blizini onih redaka i stupaca koji su bili izbrisani u prošlom koraku. Ako se ćelija nalazi pokraj izbrisanog retka, potrebno je provjeriti tvori li zajedno s ćelijom s druge strane izbrisanog retka i još nekom od dviju njima susjednih ćelija iz stupca niz triju ćelija s istom vrijednosti. U tom slučaju taj stupac označujemo za brisanje. Analogno činimo s ćelijama pokraj izbrisanog stupca.

Kao što je već navedeno, brisanje ćelija ne provodimo doslovno, nego ćemo samo pamtiti niz živih redaka i niz živih stupaca. S pomoću tih nizova po potrebi možemo pronaći za svaku ćeliju njoj susjednu te lako obrisati cijeli redak ili stupac. To su upravo one operacije koje nam trebaju da bismo mogli izvršiti potrebne operacije iz prethodnog poglavlja.

S pomoću gore opisanih observacija moguće je bilo implementirati rješenje zadatka u složenosti $O(N^2)$ što je bilo dovoljno dobro za dobivanje svih bodova na zadatku.

Primijetite da su u optimalnom rješenju vrijednosti nagrada koje je potrebno odrediti neopadajuće, tj. sve preostale nagrade možemo odrediti tako da se preostali učenici neće međusobno svađati. Zadatak možemo riješiti tako da svakog učenika kojemu je potrebno dodijeliti neku vrijednost rješavamo neovisno o drugim takvim učenicima.

Za učenika na poziciji x i neku vrijednost nagrade v možemo unaprijed izračunati:

- f(x, v) - broj učenika nakon x koji su dobili nagradu strogo veće vrijednosti od v
- g(x, v) - broj učenika prije x koji su dobili nagradu strogo manje vrijednosti od v.

Za svakog od preostalih učenika x sada možemo pronaći optimalnu vrijednost nagrade v tako da minimiziramo izraz f(x, v) + g(x, v). Složenost ovog algoritma je $O(N^2)$ što je dovoljno za osvajanje 60% bodova.

Pretpostavimo da za učenika na poziciji x imamo niz A[w] koji nam kaže s koliko bi ostalih učenika bio posvađan taj učenik ako mu dodijelimo nagradu vrijednosti w. Sve ostale učenike možemo podijeliti u dva skupa:

- Oni koji u rang listi dolaze nakon x:

  Ako je vrijednost nagrade koju je dobio učenik iz tog skupa jednaka v, tada bi se trenutni učenik posvađao s njime ako dobije nagradu vrijednosti manju od v, stoga bi interval niza A na pozicijama w = 1, 2, …, v-1 bilo potrebno povećati za 1.

- Oni koji u rang listi dolaze prije x:

  Ako je vrijednost nagrade učenika iz tog skupa jednaka v, tada bi se trenutni učenik posvađao s njime ako dobije nagradu veće vrijednosti od v, stoga bi interval niza A na pozicijama w = v+1, v+2, …, 100 000 bilo potrebno povećati za 1.

Sada optimalnu vrijednost za učenika x možemo odrediti tako da pogledamo za koji w će vrijednost A[w] biti najmanja.

Zadatak možemo riješiti tako da odredimo početni niz A za referentnog "nultog" učenika te prolazimo po učenicima redom kojim su se nalazili na rang listi prebacujući ih s prvog skupa u drugi te ažurirajući niz A[w].

Izgradnju početnog niza A obavljamo na sljedeći način: Na početku je A[w] = 0 za svaki w. Za svakog učenika koji je dobio nagradu povećamo interval u nizu A na pozicijama [1, w-1] za 1.

Nakon izgradnje početnog niza A, redom prolazimo po učenicima i radimo sljedeće:

- Ako trenutni učenik nije dobio nagradu, pronađemo minimalnu vrijednost A[w] i pripadajuću vrijednost w.

- Ako je za trenutnog učenika poznata dobivena vrijednost nagrade, tada ga mičemo iz prvog skupa i prebacujemo u drugi skup. Micanje iz prvog skupa svodi se na oduzimanje na intervalu [1, v-1] te povećavanje na intervalu [v+1, 100 000], gdje je v vrijednost nagrade koju je dobio taj učenik.

Ako bolje promotrimo operacije koje radimo na nizu A, to su sljedeće:

- Povećavanje/smanjivanje neki interval za 1.
- Određivanje najmanjeg elementa u nizu i pripadajućeg indeksa.

Takve operacije moguće je efikasno obaviti uporabom turnirskog stabla s lijenom propagacijom.

Promatrajmo najjužniju vezu između gradova M i N. Nikakve promjene na drugim vezama na nju neće utjecati. Stoga najprije "popravljamo" upravo tu vezu, tj. povećavamo joj cijenu tako da bude djeljiva s K ako je to potrebno, jer navedenu promjenu moramo učiniti bez obzira na ostale promjene. Promjenom ažuriramo sve sjeverne veze, nakon čega ispravljenu vezu između gradova M i N možemo zaboraviti jer se ona više neće mijenjati.

Nastavljamo dalje, jednako zaključujući za novu najjužniju vezu između gradova M i N - 1 (ili gradova M - 1 i N, svejedno), na koju druge promjene neće utjecati pa joj svakako moramo povećati cijenu tako da bude djeljiva s K ako je to potrebno. Ažuriramo sjeverne cijene i nastavljamo dalje na isti način, obrađujući veze od juga prema sjeveru, što se svodi na prolazak "unatrag" po M x N matrici u kojoj su zapisane cijene veza.

Ovakvo rješenje nosi samo 50% bodova jer ima veliku vremensku složenost, $O(N^4)$, budući da za svako od N^2 polja matrice ažurira i sva polja gore-lijevo (koja odgovaraju sjevernijim vezama) kojih ima $O(N^2)$. Za sve bodove potrebno je ubrzati ažuriranje veza. Stoga pri rješavanju nekog polja nećemo odmah ažurirati ostala polja. Nego, kada dođemo na neko polje (x, y), ukupnu vrijednost za koju se ono dosad povećalo računamo iz sljedeće relacije:

$$povecanje(x, y) = povecanje(x, y + 1) + povecanje(x, y + 1) - povecanje(x + 1, y + 1)$$

Ovo vrijedi zato što se svako izravno ili neizravno povećanje polja (x, y + 1) ili (x, y + 1) prenijelo i na polje (x, y), a ona povećanja koja bismo tim zbrajanjem dvaput pribrojili upravo su ona kojima se povećalo i polje (x, y + 1), pa njih oduzimamo. Uz upotrebu pomoćne matrice *povecanje*, ovo rješenje poprima složenost $O(N^2)$.

Za ovaj zadatak opisujemo dva rješenja: očekivano (suboptimalno) rješenje, kao i optimalno rješenje koje smo naknadno otkrili. Naime, očekivano rješenje zadatka, koje su imala trojica autora, kao i svi natjecatelji s visokim brojem bodova na zadatku, u nekim slučajevima daje veći broj koraka od minimalnog, čega prilikom izrade testnih primjera i bodovanja nismo bili svjesni. Poslije smo osmislili optimalno rješenje (koje opisujemo ispod), kojega nije osmislio nijedan natjecatelj, dok za natjecatelje koji su imali očekivano rješenje smatramo da su zaslužili svoje bodove. Testni primjeri rađeni su u skladu s očekivanim rješenjem i zbog toga optimalno rješenje na većini njih "pada". To znači da testni primjeri nisu u potpunosti ispravni, što je naša pogreška i zbog koje se ispričavamo.

Očekivano (suboptimalno) rješenje

Zadatak rješavamo dinamičkim programiranjem. Neka je f(x) optimalan broj koraka da sa stola odlijepimo x-ti selotejp. Razlikujemo dva slučaja:

- Selotejp x je vidljiv na površini stola na početku obavljanja operacija. Tada vrijedi f(x) = 1.
- Inače, da bi selotejp u nekom trenutku postao vidljiv na površini, najprije se moramo riješiti barem jednog selotejpa koji ga **izravno** prekriva. Neka je Y skup svih selotejpa koji izravno prekrivaju selotejp x. Tada se f(x) može izračunati prema sljedećoj formuli:

  $$f(x) = 1 + \min \{ f(y) : y \text{ element } Y \}$$

Konačno, zbog uvjeta da se prvi selotejp proteže cijelom dužinom stola, rješenje će biti sadržano u f(1).

Selotejpe koji su zalijepljeni izravno na neki drugi selotejp možemo pronaći tako da iteriramo redom po selotejpima počevši od prvog te simuliramo njihovo lijepljenje na stol tako da održavamo niz u kojem za svaki segment stola pamtimo oznaku trenutnog selotejpa koji se tamo nalazi. U 50% primjera, broj segmenata stola, D, dovoljno je malen za naivnu implementaciju.

Rješenje je moguće ubrzati korištnjem *std*::*set* ili slične strukture podataka koja će pamtiti izgled stola tako da su jedinični segmenti spojeni u cjelinu koja odgovara istom selotejpu, tj. svaki segment stola opisan je trojkom (početak segmenta, kraj segmenta, oznaka selotejpa koji trenutno prekriva taj segment). Kada na stol lijepimo novi selotejp, moguća su tri scenarija:

- Neki segment ostat će nepromijenjen (novi selotejp i promatrani segment nemaju zajedničkog presjeka).
- Neki segment bit će u potpunosti prekriven novim selotejpom - taj segment možemo obrisati iz strukture.
- Neki segment će se skratiti zbog postojanja zajedničkog presjeka s novim selotejpom - tom segmentu ažuriramo nove vrijednosti rubova segmenta.

Kada izbrišemo, odnosno ažuriramo sve potrebne segmente, možemo dodati novi selotejp u strukturu podataka. Važno je uočiti da ćemo ukupno napraviti O(N) operacija tipa 2 i O(N) operacija tipa 3, stoga je dovoljno iterirati po svim segmentima na kojima će biti potrebno obaviti neku promjenu.

Optimalno rješenje

Kao i u prethodno opisanom algoritmu, korištenjem odgovarajućih struktura možemo efikasno pronaći sve parove selotejpa koji leže izravno jedan na drugome.

Cilj nam je ukloniti najdonji selotejp. Primijetimo da selotejp možemo ukloniti ako je uklonjiv na početku ili ako smo uklonili neki selotejp koji se nalazi izravno iznad njega. Također, selotejp će biti uklonjen ako je uklonjen neki selotejp ispod njega s kojim je u izravnom kontaktu.

Možemo napraviti sljedeći težinski usmjereni graf: svaki selotejp predstavljen je jednim čvorom te za sve parove selotejpa A i B, tako da se A i B diraju pri čemu je A iznad B, dodat ćemo brid A -> B s težinom 1, te brid B -> A s težinom 0. Također, dodat ćemo poseban čvor S koji ima bridove težine 1 prema svim selotejpima uklonjivima na početku. Sada je jednostavno provjeriti da je traženo rješenje težina najkraćeg puta od S do najnižeg selotejpa.

Primijetimo da je zadatak ekvivalentan sljedećemu: potrebno je odabrati neparan broj K i K permutacija brojeva od 1 do N tako da za M zadanih uređenih parova (A, B) vrijedi da se A pojavljuje nakon B u više od pola permutacija. Pritom k-ta permutacija predstavlja relativan poredak snaga k-tog najjačeg igrača iz timova.

Za dani niz permutacija, neka *f(A, B)* označava razliku broja permutacija u kojima se A pojavljuje prije B i broja permutacija u kojima se B pojavljuje prije A. Ideja je sljedeća: želimo pronaći mali broj permutacija kojima ćemo promijeniti vrijednost *f(A, B)* za neke parove (A_1, B_1), (A_2, B_2), … (A_s, B_s) na željeni način, a da se pritom ne promijeni vrijednost funkcije f za ostale parove. Koliko smo uspješni da s malim brojem permutacija "riješimo" velik broj parova, ovisit će o broju bodova koji algoritam nosi. Vidjet ćemo pritom da će naši algoritmi napraviti 200 permutacija, tj. 200 igrača, ali da će *f(A, B)* biti barem 2 za tražene parove. Tada ćemo izbacivanjem proizvoljne permutacije dobiti traženo rješenje.

Kako bismo opisali algoritam do kraja, uvedimo sljedeću notaciju: za dani skup A neka R(A) označava niz brojeva iz A u rastućem poretku, a P(A) neka označava niz brojeva iz A u padajućem poretku. Također, prisjetimo se notacije za skupovnu razliku: (A / B) označava skupovnu razliku skupova A i B, tj. skup koji sadrži sve brojeve koji se nalaze u skupu A, a ne nalaze se u skupu B. Također označimo s S skup svih brojeva od 1 do N.

Opišimo rješenje koje pravi dvije permutacije po svakom zadanom paru, što će izbacivanjem jedne davati rješenje vrijedno 30% bodova. Za par (A, B) dodat ćemo permutacije:

(R(S \ {A, B}), A, B)

(A, B, P(S \ {A, B})).

Primjerice, za N = 5 te A = 2, B = 5, dane permutacije bile bi:

(1, 3, 4, 2, 5)

(2, 5, 4, 3, 1)

Primijetite da se pritom *f(A, B)* povećao za 2, a *f* se nije promijenila ni za jedan drugi par čime smo rješenje ako jednostavno napravimo po dvije opisane permutacije za svaki uređeni par iz ulaza.

Za još 30% bodova potrebno je napraviti najviše četiri permutacije za svaki broj od 1 do N.

Neka je odabrani broj x, te neka je A skup svih brojeva tako da je (A, x), uređeni par u ulazu (tj. A je skup onih brojeva a za koje želimo da vrijedi *f(a, x)* > 0), a B = S \ A \ {x}, tj. skup svih ostalih brojeva različitih od x. Ove četiri permutacija dat će nam traženo rješenje:

(P(A), x, P(B))

(R(A), x, R(B))

(x, P(B), P(A))

(R(B), R(A), x)

Za 100% bodova, podijelit ćemo sve neuređene parove brojeva u najviše 100 skupina tako da se svaki broj pojavljuje u najviše jednom paru u skupini. Za početak možemo pretpostaviti da je N = 100, jer u suprotnom ćemo izgraditi "veći" turnir od 100 ekipa pri čemu možemo jednostavno ne ispisati sve njih.

Traženu podjelu parova na skupine naći ćemo na sljedeći način: promatrajmo potpuni neusmjereni graf od 101 čvorova. Podijelit ćemo njegove bridove na 50 Hamiltonovih ciklusa tako da se svaki brid nalazi u točno jednom Hamiltonovom ciklusu. Neka su vrhovi čvorova označeni brojevima od 0 do 100.

Ovaj problem moguće je riješiti u većoj općenitosti (za više detalja pogledajte zadatak Hyperloop). S obzirom da je 101 prost broj, postoji i jednostavnije rješenje. Naime sljedeći niz Hamiltonovih ciklusa zadovoljava teražene uvjete:

(0, k mod N, 2k mod N, 3k mod N, … (N-1)k mod N) za sve k od 1 do 50.

Uzimajući svaki drugi brid te ignorirajući pomoćni čvor s indeksom 0, dobit ćemo dvije skupine bridova za svaki Hamiltonov ciklus i svih 100 skupina zadovoljavat će traženi uvjet.

Preostaje za dani skup uređenih parova (A_1, B_1), (A_2, B_2), … (A_s, B_s), pri čemu se nijedan broj ne pojavljuje dvaput, konstruirati dvije permutacije koje će promijeniti samo *f(A, B)* za dane parove. Neka je T skup svih brojeva koji nisu ni u jednom paru. Tražene su permutacije:

($A_1, B_1, A_2, B_2, \ldots A_s, B_s, R(T)$)

($P(T), A_s, B_s, \ldots A_2, B_2, A_1, B_1$).

Zadatak je moguće riješiti i na druge (jednostavnije) načine. Primjerice, jedan natjecatelj osvojio je 100% bodova induktivnim pristupom.